Curve fitting software for first order plus dead time (FOPDT) model parameter estimation using step or pulse response data: a tutorial

Kevin Burn[a*], Chris Cox[a]

[a] *Department of Computing, Engineering and Technology, University of Sunderland, SR6 0DD, United Kingdom.*

[*]Corresponding author.
Email: kevin.burn@sunderland.ac.uk, Tel: +44 191 515 2778

Abstract

First order plus dead time (FOPDT) models are used to represent the behaviour of many processes in the physical world. This is because the response of such models to step or pulse inputs can offer a good approximation to the actual response of many systems and sub-systems found in industrial process control. In such cases the system identification problem is simplified to that of parameter estimation. However, although the theoretical development of many system identification algorithms is available in the literature, researchers rarely release developed code for third-party use. This paper describes a modern FOPDT system identification algorithm based upon an integral equation (IE) method and demonstrates how it can be implemented within any appropriate programming environment. It also describes how existing curve fitting software developed for the numerical computing environment MATLAB can be used to estimate the FOPDT parameters and step-by-step tutorials for using three different methods are presented. The different techniques are compared subjectively and tested using simulated data. Finally they are used to identify a laboratory process using experimentally obtained data.

Keywords

System identification; parameter estimation; step and pulse response; curve fitting; MATLAB; Curve Fitting Toolbox; Optimization Toolbox; EzyFit Toolbox.

1. Introduction

For the design of many control algorithms, a process model is required. When the control algorithm is of the proportional plus integral (PI) or the proportional plus integral plus derivative (PID) form, the process model is often assumed to have a first order plus dead time (FOPDT) structure. This is because a FOPDT response is a good approximation to the actual response of many systems and sub-systems found in industrial process control applications. In addition, tuning of the PID controller parameters can be initiated using formulae that specify their values in terms of the FOPDT parameters. Popular tuning methods include those attributed to Ziegler-Nichols [1], Cohen-Coon [2] and Kappa-Tau [3]. This paper concentrates on the estimation of FOPDT model parameters using step and pulse response data.

Early approaches to system identification relied on graphical techniques and the methods of Ziegler and Nichols [1], Oldenbourg and Satorious [4], Sten [5] and Rake [6] epitomize these methods. For monotonic step responses, the collection of approaches that are based on specific area calculations are well represented in the paper of Nishikawa *et al.* [7] and the book by Astrom and Hagglund [8].

More recently, a family of new methods has been published that is more computationally involved but allows the parameters of the transfer function model, including the time delay, to be estimated simultaneously. They are commonly referred to as integral equation (IE) methods. The original ideas of Bi *et al.* [9] and Wang and Zhang [10] have been extended to allow identification under transient initial conditions [11]. Other developments have also been reported. For example, in references [12] and [13], new methods to handle 'piecewise step tests' have been presented.

In all of the above cases special purpose software has been developed that is generally not freely available. However, an alternative approach, using more readily available, general purpose, proprietary curve fitting software, has the potential to provide a simpler solution to the determination of FOPDT model parameters for those who do not have access to the kind of specialist software described in the research literature.

For a number of years there has been an interest in the relationship between system identification and curve fitting [14]. The aim of this paper is to illustrate how system identification can be performed as a curve fitting exercise. After consideration of a recent IE system identification method [11], three curve fitting software solutions available for the MATLAB software environment are employed to estimate the parameters of a FOPDT model using data generated under simulation. Detailed instructions on how to employ the various methods are given and an assessment of their accuracy and ease-of-use is made. The IE method and the best of the curve fitting methods are then employed on experimental data acquired in the laboratory and a further comparison made.

The paper is structured as follows. In Section 2 the theory behind the IE method is presented, for both step and pulse input data. Detailed pseudocode showing how to implement the method for a step input using any suitable software platform is also shown. Section 3 introduces three curve fitting techniques available in MATLAB, which are presented in a tutorial style to help readers repeat the tests with appropriate data. It concludes with a brief evaluation of the advantages and disadvantages of the different methods. In Section 4, the IE method and one of the curve fitting methods are used to fit a FOPDT model to (i) a simulated higher order system, and (ii) experimental data obtained from a laboratory water level control system. Conclusions and recommendations for further work are discussed in Section 5.

2. FOPDT parameter estimation using the integral equation (IE) method

2.1 Using step response data

The model of the system under consideration is assumed to be represented by the transfer function:

$$G_p(s) = \frac{K \exp(-sD)}{Ts + 1} \tag{1}$$

In equation (1), K is the process gain, T is the time constant, and D is the dead time, or time delay, as it will be referred to henceforth. Now consider the process response $y(t)$ to a

step input *u(t)* from 0 to *h*, at some time $t > 0$. If the system is initially at rest the response will be of the form shown in Fig. 1.

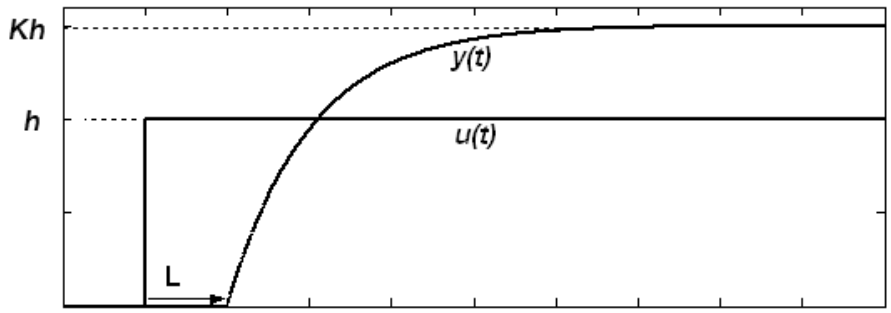

Fig. 1. Step response of a first order plus dead-time (FOPDT) model

Assuming the initial conditions to be zero, the basic equation can be re-written in a form more amenable to the analysis under discussion.

$$TsY(s)+Y(s)=KU(s)\exp(-sD)+E(s) \tag{2}$$

In equation (2), E(s) represents any error present due to noise, nonlinearity, or that arising from any other inaccuracies in the model.

Equation (2) has three unknowns: *T*, *K* and *D*. A paper by Ahmed *et al.* [11] states that an equation allowing simultaneous estimation of the unknowns can be obtained firstly integrating equation (2). The result of this single integration is shown in equation (3):

$$\frac{Y(s)}{s}=-TY(s)+K\frac{U(s)}{s}\exp(-sD)+E_I(s) \tag{3}$$

Taking inverse Laplace transforms of equation (3) yields:

$$y^{[1]}(t)=-Ty(t)+Ku^{[1]}(t-D)+e_I(t) \tag{4}$$

where the first order integrals of *y(t)* and *u(t)* are defined as follows:

$$y^{[1]}(t)=\int_0^t y(\tau)d\tau\,,\ u^{[1]}(t)=\int_0^t u(\tau)d\tau \tag{5}$$

Equation (4) is valid for any bounded input signal *u(t)*. Notice that D remains an implicit parameter that cannot be directly estimated. When the input is a step function, the problem is overcome as follows.

It has been shown in [11] that if the input is a step function of height h applied at $t = 0$, the following integral holds for $t \geq D$:

$$u^{[i]}(t-D)=\frac{h(t-D)^i}{i!} \tag{6}$$

For a step input the estimation equation (4) can be written for $t \geq D$:

$$y^{[1]}(t) = -Ty(t) + Kh(t - D) + e_1(t) \tag{7}$$

This can be expanded and re-arranged to give:

$$y^{[1]}(t) = -Ty(t) + Kht - KhD + e_1(t) \tag{8}$$

Equation (8) can be expressed in least-squares estimation form as follows:

$$y^{[1]}(t) = \begin{bmatrix} -y(t) & t & 1 \end{bmatrix} \begin{bmatrix} T \\ Kh \\ -KhD \end{bmatrix} + e_1(t) \tag{9}$$

Using a more general terminology, equation (9) can be written in the form

$$\gamma = \phi(t)\theta + e(t) \tag{10}$$

where $\gamma = y^{[1]}(t)$, $\phi = \begin{bmatrix} -y(t) & t & 1 \end{bmatrix}$, $\theta = \begin{bmatrix} T \\ Kh \\ -KhD \end{bmatrix}$, and *e(t)* is noise.

Using all of the sampled values of $y^{[1]}(t)$, *y(t)* and *t*, equation (10) can be written as follows:

$$\Gamma = \Phi\theta + \xi \tag{11}$$

where

$$\Gamma = \begin{bmatrix} \gamma_1(t_d + 1) \\ \gamma_2(t_d + 2) \\ \gamma_3(t_d + 3) \\ . \\ . \\ \gamma_1(t_N) \end{bmatrix}, \quad \Phi = \begin{bmatrix} \phi_1(t_d + 1) \\ \phi_2(t_d + 2) \\ \phi_3(t_d + 3) \\ . \\ . \\ \phi_1(t_N) \end{bmatrix} \tag{12}$$

In equation (12), *N* is the total number of samples and *d* is the delay expressed in number of sampling intervals, so that $D = d \cdot \Delta t$, where Δt is the sample time.

The least squares estimation $\hat{\theta}$ of θ in equation (12) is given by:

$$\hat{\theta} = \left(\phi^T \phi\right)^{-1} \phi^T \Gamma \tag{13}$$

This enables *K*, *T* and *D* to be calculated from the three rows of $\hat{\theta}$ i.e. $T = \hat{\theta}[1]$, $K = \hat{\theta}[2]/h$ and $D = -\hat{\theta}[1]/(Kh)$.

For a noise-free FOPDT process modelled by a FOPDT transfer function, perfect values (i.e. 0% error) are obtained for the parameters, provided $\hat{D} \geq D$. With data obtained from real systems, however, noise usually has a detrimental effect, resulting in biased values for the estimated parameter values. One method of minimizing the effects of noise is to use the instrumental variable least-squares method [9, 15-17]. This is briefly summarised as follows. Even though the measurement noise is assumed white, the integration operation performed on *y(t)* results in a coloured error term, which culminates in the parameter values being biased. To counteract this, the IV method uses a surrogate system with 'guessed' parameters, with the same input as the real system, but which is not influenced by noise. How this is achieved in practice is shown in the following section.

2.2 Algorithm implementation

Details of the algorithm to implement the IE method in software are presented below in the form of pseudocode. This is shown in Algorithm 1, where the following points should be noted:

- The pseudocode is intended to be language independent, so that it can be implemented with any appropriate programming language/platform. For this work, the authors used a MATLAB m-file.
- All quantities in bold are either vectors or matrices.
- The algorithm requires time, input, and output data from a step test as vector parameters (***t***, ***u*** and ***y***, respectively).
- The sampling time T_s is known and constant, and there are *N* samples in each of the ***t***, ***y*** and ***y*** vectors, e.g. ***u*** = *u[1], u[2], u[3], ..., u[N] etc.*
- The estimate for the time delay used in the iteration process is actually relative to time *t=0*. Hence the actual time delay is *D* – (step time). This is given the symbol *L* in Algorithm 1.
- The variable for the previous loop estimate of *D*, expressed as an integer multiple of Ts (*D_guess_z*), is initially set high in order to ensure the algorithm enters the iteration loop during the first pass (line 6).

Algorithm 1. INTEGRAL EQUATION METHOD(***t***, ***u***, ***y***)

1	$h = u[N] - u[1]$	// Step size
2	$T_s = t[2] - t[1]$	// Sample time
3	$Tstep = max(u[i+1] - u[i]),\ i = 1,N$	// Time at which step input occurs
4	$\boldsymbol{y1} = integral(\boldsymbol{y})$	// Numerical integration
5	$D\_guess = T_s$	// Initial guess for (*L* + *Tstep*) used in // iteration loop
6	$D\_guess\_z = 999$	// Previous estimate for *D_guess*
7	$D\_count = ceiling(D\_guess / T_s)$	// D_guess expressed in terms of number // of samples
8	$count = 1$	// Loop counter
9	$while \left\lvert Dguess - Dguess\_z \right\rvert > T_s$	//Convergence condition
10	$\boldsymbol{t_m} = t[(dd+1)...N_t]$	// Redefine t, y, y1 data so that it starts
11	$\boldsymbol{y_m} = y[(dd+1)...N_t]$	// at sample *(dd+1)*

12 $\boldsymbol{y1_m} = y1[(dd+1)...N_t]$

13 $\boldsymbol{phi} = [-\boldsymbol{y_m}\ \boldsymbol{t_m}\ \boldsymbol{1}]$ // Least squares estimation

14 *if count == 1* // First loop only...

15 $\boldsymbol{psi} = \boldsymbol{phi}$

16 *else*

17 $\hat{y}_m = \hat{y}[(dd+1)\cdots N]$ // Estimate, calculated below

18 $\boldsymbol{psi} = [\hat{y}_m\ \boldsymbol{t_m}\ \boldsymbol{1}]$

19 *end if*

20 $\theta = (\boldsymbol{psi'}*\boldsymbol{phi})^{-1} * \boldsymbol{psi'} * \boldsymbol{y1_m}$ // Equation (11)

21 $T = \theta[1]$ // Calculate *T*, *K* and *L*

22 $K = \theta[2] / h$

23 $D = -\theta[3] / (K * h)$

24 $L = D - tstep$

25 *if* $L < 0$ $L = T_s$ // Prevents negative L causing instability

26 $G_m = Ke^{-sL}/(Ts + 1)$ // Surrogate system

27 $\hat{y} = L^{-1}[Gm(s).U(s)]$ // Inverse Laplace transform to get y
// estimate

28 *Dguess_z = Dguess;* // Update previous estimate

29 *Dguess = round(D*100)/100;* // Accurate to within one sample period

30 *dd=ceil(Dguess/Ts);* // Number of samples of *Dguess*

31 *count = count + 1;* // Update loop counter

32 *end* // Of while loop

The MATLAB m-file implementation of Algorithm 1 can be obtained from the authors if a request is made by email.

2.3 Using pulse response data

Although a step function is accepted by many as an appropriate input for system identification, there are situations where an input with a less disruptive response is desirable. Such inputs include a simple pulse, characterised here as having an amplitude *h* and width *W*. For analysis purposes a 'pulse' can be considered as a summation of delayed steps and can be expressed as:

$$u(t) = \sum_{i=0}^{1} h_i (t - L_i)\Omega(t - L_i) \tag{14}$$

where

$$\Omega(t - L_i) = \begin{cases} 0 \text{ for } (t - L_i) < 0 \\ 1 \text{ for } (t - L_i) \geq 0 \end{cases} \tag{15}$$

In the case of the simple pulse, $W = L_1 - L_0$; $h_1 = -h_0$.

For the input defined by equation (14), the delayed signal can be expressed as:

$$u(t-D)=\sum_{i=0}^{k} h_i(t-L_i-D)\Omega(t-L_i-D) \tag{16}$$

If $$\Omega_i=\Omega(t-L_i-D) \tag{17}$$

Then (16) can be simplified to:

$$u(t-D)=\sum_{i=0}^{k} h_i(t-L_i-D)\Omega_i \tag{18}$$

It follows from (4) and (18) that:

$$y^{[1]}(t)=-Ty(t)+K\sum_{i=0}^{1} h_i(t-L_i-D)\Omega_i+e_I \tag{19}$$

Expanding (19):

$$y^{[1]}(t)=-Ty(t)+K\sum_{i=0}^{1} h_i(t-L_i)\Omega_i-KD\sum_{i=0}^{1} h_i\Omega_i+e_I \tag{20}$$

Finally, expressing the result of equation (20) into the form of equation (10) leads to:

$$\gamma_1=y^{[1]}(t),\ \ \phi_1=\begin{bmatrix}-y(t) & \sum_{i=0}^{1} h_i(t-L_i)\Omega_i & \sum_{i=0}^{1} h_i\Omega_i\end{bmatrix},\ \theta_1=\begin{bmatrix}T\\K\\-KD\end{bmatrix} \tag{21}$$

It is a relatively straightforward task to modify Algorithm 1 for a pulse input.

3. Curve fitting using MATLAB

3.1 Introduction

Norton [18] describes identification as the process of constructing a mathematical model of a dynamical system from observations and prior knowledge. In a report produced by the Open Sustainability Technology Lab of Michigan Tech, curve fitting is described as the determination of a mathematical model that best fits experimental data [19]. In addition, their best fit is quantitatively defined as the minimisation of the difference between the data and the curve. Finally, in Ljung [14] formalised the ‘kinship’ between the identification of linear dynamical systems and classical curve fitting.

The ‘best fit’ problem for experimental data frequently involves a least squares solution. Often these problems are linear, but in the case of a FOPDT response it is nonlinear because the curve to be used in the fitting process is not linear in the parameters. In such cases it is necessary to resort to nonlinear least squares methods that involve an iterative improvement to the parameter values. In principle this is similar to the iterative loop employed in the IE

method described in section 2. An excellent article by Gavin [20] provides a detailed analytic treatise of one of the more popular strategies, known as Levenberg-Marquardt algorithm (LMA). However, the approach here is not to describe the mathematical basis of these various methods, but to explain how the various software solutions that are based on these algorithms can be exploited.

In Ljung's paper [14], the response of a linear dynamical system in continuous time is described by:

$$y(t) = G(p)u(t) + v(t) \tag{22}$$

where p is the differentiation operator. In the general identification problem the aim is to find transfer operator $G(p)$ and possibly the spectrum of the additive noise $v(t)$. However, in this case the structure of the transfer function is known to have the form described by equation (1). Consequently, the following can also be shown.

For a step function defined by:

$$u(t) = \begin{cases} 0 \text{ for } t < 0 \\ 1 \text{ for } t \geq 0 \end{cases} \tag{23}$$

The response is given by:

$$y(t) = Kh\left[1 - \exp\left(-(t-D)/T\right)\right] \cdot H\left[t - D\right] \tag{24}$$

where $H[t - D]$ is the Heaviside step function. For a pulse of height h and width W:

$$y(t) = \begin{cases} 0 & t < 0 \\ h \quad \text{for} & 0 < t \leq W \\ 0 & t > W \end{cases} \tag{25}$$

$$y(t) = Kh\left[1 - \exp\left(-(t-D)/T\right)\right] \cdot H\left[t - D\right] - Kh\left[1 - \exp\left(-(t-D-W)/T\right)\right] \cdot H\left[t - D - W\right] \tag{26}$$

It is thus equations (24) and (26) that will represent the theoretical 'curves' that are to be fitted to the experimental data, with K, T and D the parameters to be identified.

The following sections describe the use of three proprietary software applications, developed for use within a MATLAB environment: the Curve Fitting Toolbox, the Optimization Toolbox, and a free curve-fitting toolbox called EzyFit. The information is presented in an active and direct style, in order to facilitate its use as a tutorial for the new user.

For each of the methods described it is assumed that step or pulse response data (as appropriate) is available in the MATLAB workspace, in the form of time and output vectors (usually assigned symbols t and y, respectively. The data used in the following sections was generated using a Simulink simulation, via *To Workspace* blocks with the *Save format* set to *Array*.

3.2 Curve Fitting Toolbox

The Curve Fitting Toolbox can be used in two different environments: via a graphical user interface (GUI) (which is the one followed below), or via the MATLAB command line. This section provides an overview of those parts of the toolbox which are required to identify a FOPDT transfer function using noise-free data generated by a Simulink model. For those wishing to have a more complete understanding of all of the features available when using the toolbox, a good reference is that provided by the support documentation [21].

To illustrate the basic methodology consider the problem of determining the parameters of a FOPDT model characterised by the transfer function:

$$G_p(s) = \frac{1.25\exp(-2.15s)}{2s+1} \tag{27}$$

To open the curve fitting tool, enter *cftool* at the prompt in the command window. This opens the Curve Fitting Tool window shown in Fig. 2.

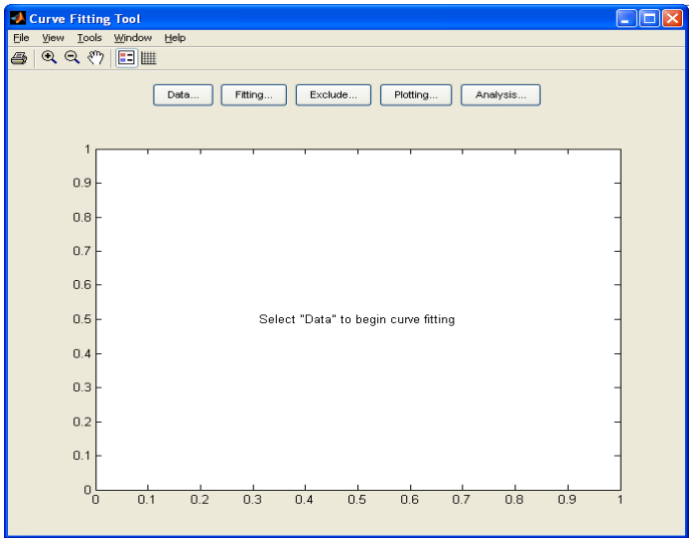

Fig. 2. The Curve Fitting Tool Window

The next step is to click on the *[Data...]* button in shown in Fig. 2, which opens the Data window, similar to that shown in Fig. 3 but with the Preview pane initially empty. This window allows X Data and Y Data to be imported from the workspace (i.e. *t* and *y* data for this application). Once the data has been selected via the drop-down list, a preview of the data that will be used for the identification phase appears, as shown. To proceed to the next window, click on the *[Create data set]* button.

The original window (Fig. 2) now includes the preview data. The next step is to click on the *[Fitting...]* button. This reveals the Fitting window, similar to that shown in Fig. 4, but with the *Custom* Equations and Results panels initially empty. Clicking on the *[New fit]* button allows the user to give the session a name (*Fit name*) and choose a *Type of fit*. Since the *Exponential* library does not contain the desired equation, the user must create their own using the *Custom Equations* alternative. This opens the New Custom Equation window shown in Fig. 5.

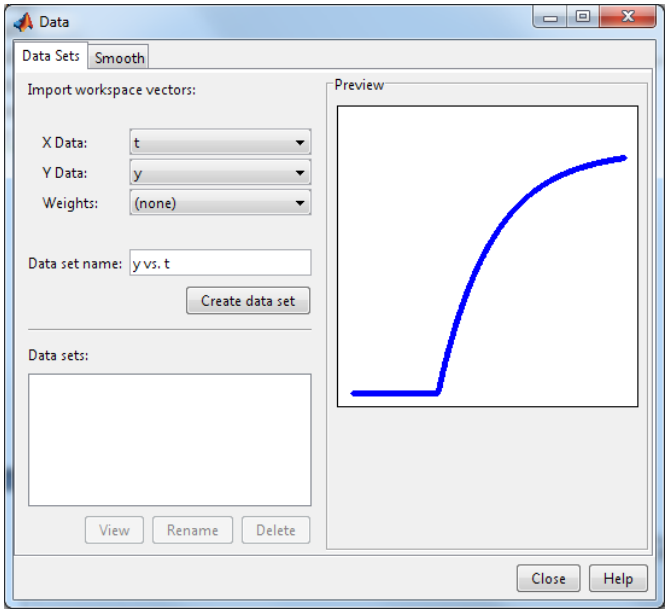

Fig. 3. The Data window

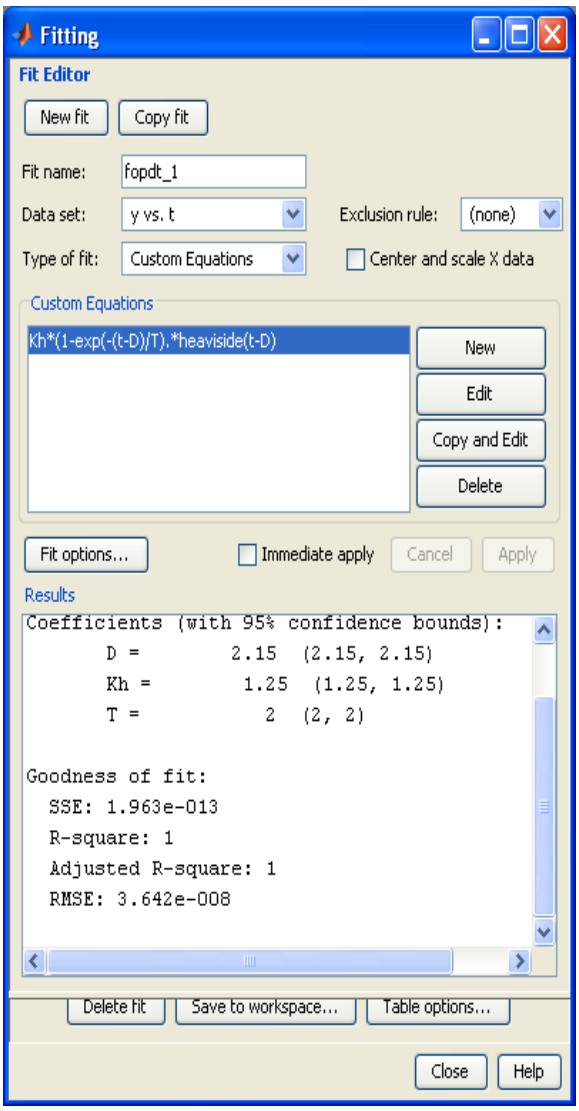

Fig. 4. The Fitting window

Firstly, choose the General Equations tab, change the independent variable to *t*, then type in the required equation. In MATLAB syntax the FOPDT response is defined as follows:

$$y = Kh*(1-\exp(-(t-D)/T)).*heaviside(t-D) \qquad (28)$$

In equation (28), *heaviside(t-D)* is (unsurprisingly) the Heaviside function. This is implemented in the MATLAB Symbolic Toolbox. However, if this toolbox is not installed, it is a relatively simple matter for anyone with even rudimentary skills in writing MATLAB m-files to develop a user-written equivalent.

Clicking on the *[OK]* button produces the response shown in Fig. 6. In addition, the Results pane in the Fitting window displays the values for *Kh*, *T* and *D*, together with some well- known characteristics that define the quality or goodness of fit (Fig. 4). In the case shown here the date was ideal and noise-free and the model chosen is the right one. Thus, the results were exact, as shown in Fig. 5, i.e. *K = 1.25*, *D = 2.15*, *T = 2*. In Fig. 5 the blue diamonds represent the experimental data and the red curve is derived using the values displayed for *Kh*, *T* and *D*.

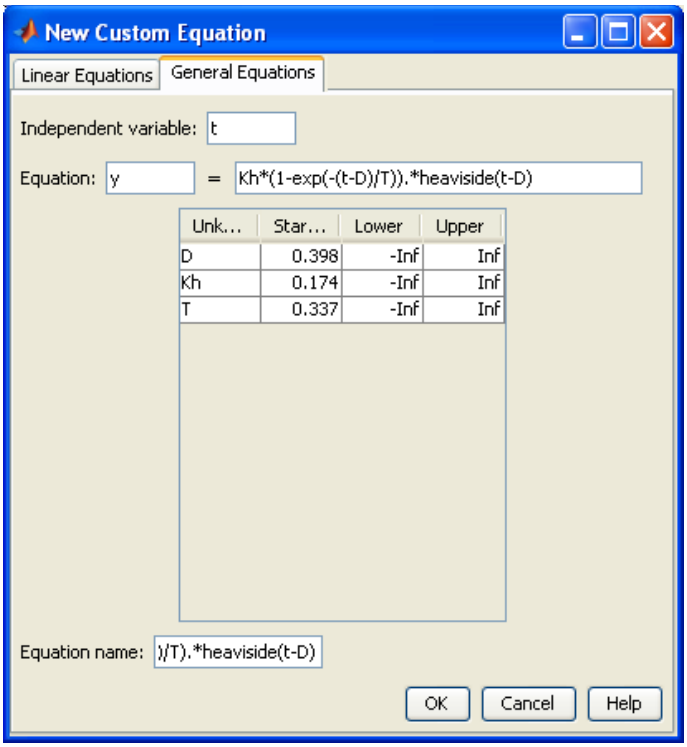

Fig. 5. New custom equation window

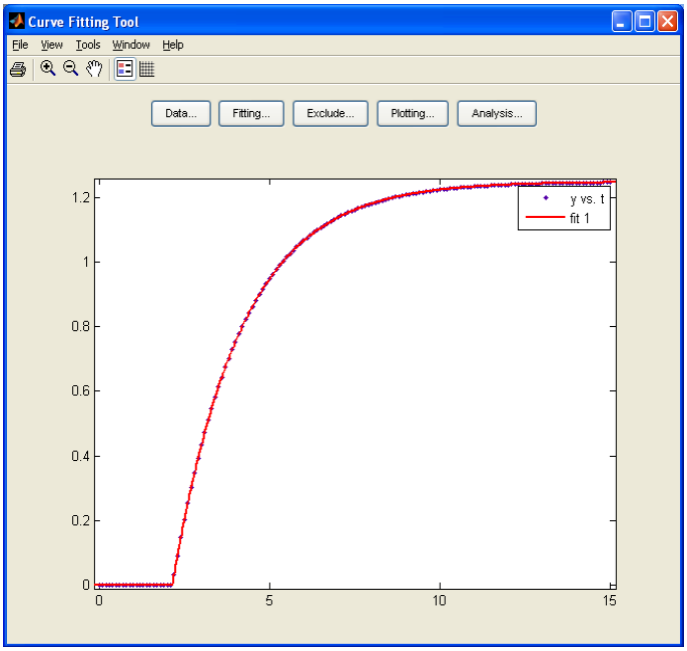

Fig. 6. The final window

3.3 Optimization Toolbox

For MATLAB users who do not have access to the Curve Fitting Toolbox but do have access to the Optimisation Toolbox [22], an alternative solution to the system identification problem is possible to exploit the *lsqcurvefit* algorithm.

The *lsqcurvefit* function solves nonlinear least squares problems and is essentially a convenient interface for data-fitting problems. It uses the *lsqnonlin* algorithm, which is the same as that used by the Curve Fitting Toolbox. In the case of the FOPDT transfer function it enables the user find values for *K*, *T* and *L* comprising the vector *a* which solve the problem:

$$\min \sum \left[F(a, XData) - YData \right]^2 \tag{29}$$

given an input data vector *XData* and an observed data vector *YData*, both of length N. In addition, *F(a, XData)* is a vector valued function that captures the step or pulse responses, as presented earlier. Note that in this case, the input pulse must begin at a time greater than zero i.e. $t > 0$, or the algorithm described below fails and an error message is generated. The normal m-file formulation makes use of the anonymous function concept. An anonymous function is one that is not stored in the program, but one that accepts an input and returns an output, just as standard functions do. In addition, it contains a single executable statement. As shown in the MATLAB m-file below, the @ operator creates the anonymous function.

For a single pulse input, of height H and width W, the FOPDT curve fitting m-file has the following form (Algorithm 2):

Algorithm 2. PULSE INPUT IDENTIFICATION USING OPTIMIZATION TOOLBOX

```
function [a,resnorm]=optlsqFOPDTpulse(W,H,Working_Data);
      t=Working_Data(:,1);
      y=Working_Data(:,2);
      f=@(a,t)a(1)*H*(1-exp(-(t-a(3))/a(2)))...
```

```
            .*heaviside(t-a(3))-a(1)*H*(1-exp(-(t-a(3)-W)/a(2)))...
            .*heaviside(t-a(3)-W);
      [a,resnorm] = lsqcurvefit(f,[0.5 0.5 0.5],t,y)
      K=a(1); T=a(2); D=a(3);
end
```

For the transfer function given by equation (27), the results produced by Algorithm 2 for a sample time $T_S = 0.1$ were *K = 1.2497*, *D = 2.1603*, *T = 1.999.*, with *resnorm = 5.0072e-007*. Perfect results were obtained using the IE method with a pulse response.

3.4 EzyFit Toolbox

The EzyFit Toolbox is a free add-on for MATLAB, which was intended to offer a simple and efficient way to perform curve fitting with a range of nonlinear fitting functions [23]. The web site provides a wide range of details about the software including system requirements, installation instructions and a demonstration example to illustrate the methodology and provide further information about the software.

After downloading and installing the toolbox, and adding the necessary input/output (*t*, *y*) data in the workspace, the first step is to type at the MATLAB prompt:

```
>> efmenu
```

Assuming the MATLAB workspace includes (*t*, *y*) data, it should then be plotted using the standard MATLAB command:

```
>> plot(t,y)
```

This results in the appearance of a standard MATLAB plot of the response. However an additional option appears to the right hand side of the menu: *EzyFit*. Selecting this option produces a window similar to Fig. 7. Clicking on *Show fit* reveals a list of different kinds of functions. However none of the standard exponential functions are suitable for modelling a FOPDT response. Consequently, it is necessary to click on *Edit User Fit*. This produces a new list similar to that shown in Fig. 7. If the *New User Fit...* option is selected an Edit User Fit window opens that allows the user to customise the fit, as shown in Fig. 8. The new list shown in Fig. 8 (*#1: myfit*; *#2: fit3* etc.) are all user-defined functions. In the following discussion the function designed here was named *csc1*.

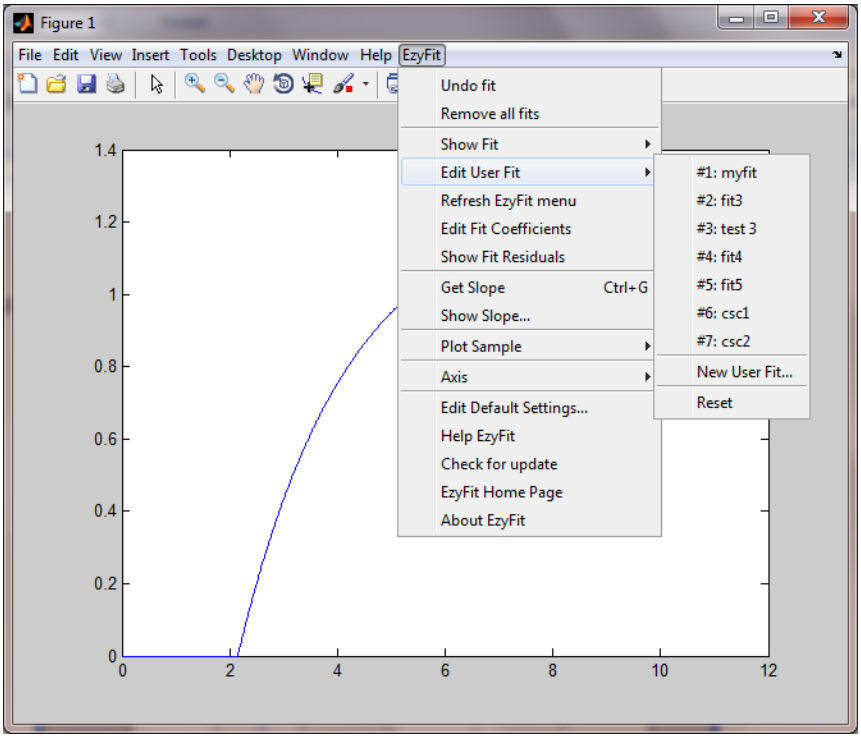

Fig. 7. EzyFit menu

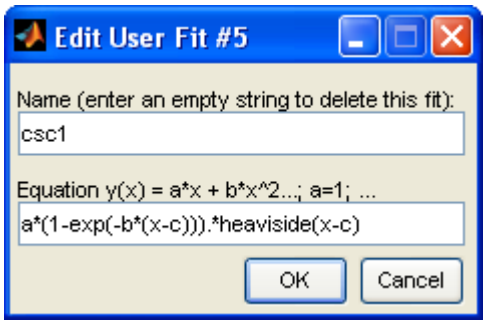

Fig. 8. Edit User Fit window

After returning to the original plot window, selecting *Show* Fit and *csc1* reveals the results shown in Fig. 9. Note that $T = 1/b = 2$. Once again, almost perfect estimates for the FOPDT parameters have been obtained.

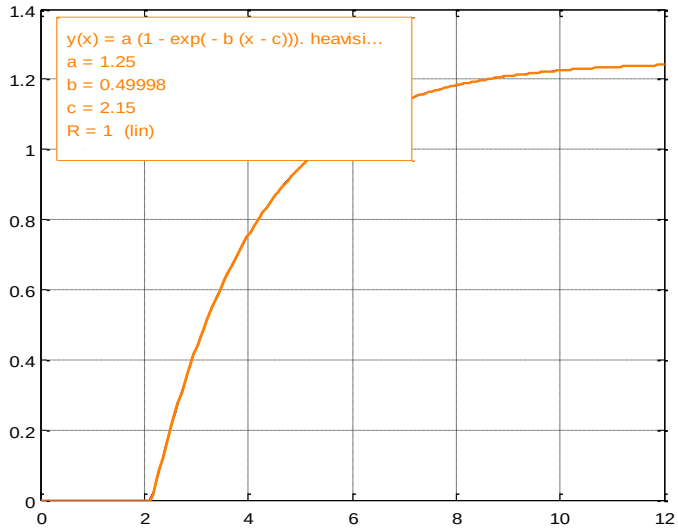

Fig. 9. EzyFit results

3.5 Comparison of the different methods

For the step response tests, essentially perfect (i.e. error-free) results were obtained using the IE method, the Curve Fitting Toolbox and the EzyFit Toolbox. For the pulse response tests, perfect results were obtained using the IE method, but small errors were recorded when using the Optimization Toolbox. However the greatest error, which was in the estimation of the time delay D, was less than 0.5%.

Hence, when using idealised data, the performance of the various methods can hardly be separated in a quantifiable sense. This is not surprising given that three of the methods – IE, Curve Fitting Toolbox and EzyFit Toolbox – are all based upon a least squares approach. The main differences between the methods lay in their relative ease of use and required prior knowledge to apply them.

Due to the relative complexity of using the Optimization Toolbox compared to both the Curve Fitting Toolbox and EzyFit, it was decided that it would not be used during further testing, as described in the next section. The principal advantages and disadvantages of the other approaches can be summarised as follows.

Integral equation (IE) method

Advantages:

- It can be developed as a standalone application, not tied to any computing platform or programming language.
- The underlying equations can be developed for higher order systems, systems with transfer function zeros, non-minimum phase systems, systems with transient initial conditions, and different inputs (step, pulse, ramp etc.)

Disadvantages:

- Implementation requires advanced programming skills.
- Modification of the FOPDT equations requires an understanding of the underlying theory and mathematics.

Curve Fitting Toolbox

Advantages:

- It is easy to learn.
- It can be run from a GUI-based user interface.
- A wealth of supporting information is freely available online.

Disadvantages:

- The Toolbox must be purchased and the correct version used within a MATLAB environment.
- The FOPDT structure is not a default, user-selectable option.

EzyFit Toolbox

Advantages:

- It is free to download and install, and there is no time limit on its use.
- It is easy to use.

Disadvantages:

- It must be used within a MATLAB environment.
- The FOPDT structure is not a default, user-selectable option.

4. More realistic examples of FOPDT modelling

A large number of industrial processes with monotonic step responses can be modelled by a FOPDT transfer function of the form defined by equation (1) [24]. In addition, there is widespread use of the model in the design of PI and PID controllers. For example in the book by O'Dwyer [25], tuning rules for PI/PID controllers proposed over six decades are presented, many of which assume the availability of a FOPDT model.

This section considers two different situations that deviate from the ideal FOPDT data considered until now. The first is where the transfer function of the system to be identified is of order greater than one. The second concerns a laboratory process with an unknown transfer function; however, even in its simplest mathematical representation, the characterising equations are nonlinear. In addition, the measured response is contaminated by noise.

In both cases, the aim is to determine FOPDT models from step and pulse inputs, using the IE method, the Curve Fitting Toolbox, and the EzyFit Toolbox, as appropriate.

4.1 Simulated higher order process

The chosen model was third order and defined by the following transfer function:

$$G_p(s) = \frac{2\exp(-0.5s)}{(s+1)^3} \tag{30}$$

Step response. Data was acquired from a Simulink model of the transfer function given by equation (30), to which a unit step was applied at t = 1s. Twelve seconds of data was collected with a sampling time of 0.01s. The results of the three identification methods are shown in Table 1.

Table 1
Third order system, step response: FOPDT parameters

Identification method	K	T	L
IE method	2.046	2.124	1.563
Curve Fitting Toolbox	2.057	2.226	1.557
EzyFit Toolbox	2.074	2.283	1.543

The response of the system compared to that FOPDT models identified using the IE and the EzyFit Toolbox is shown in Fig. 10. For clarity the model identified by the Curve Fitting Toolbox has been omitted.

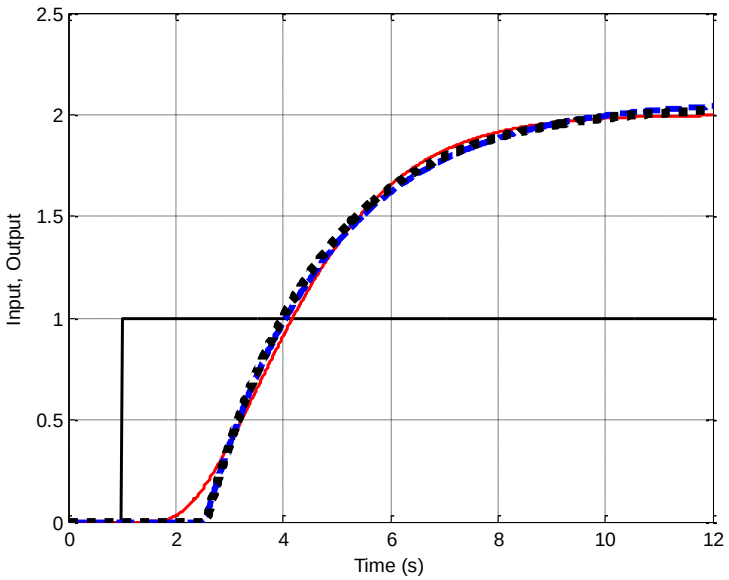

Fig. 10. Unit Step responses: Solid = Input/3rd Order response; Dotted = IE model; Dashed = EzyFit Toolbox model

Pulse response. In the second example the step function was replaced by a single pulse of width 4 seconds and height 1 unit. The model derived using the IE method was compared with that obtained from the Curve Fitting Toolbox. The results are shown in Table 2.

Table 2
Third order system, pulse response: FOPDT parameters

Identification method	K	T	L
IE method	2.046	2.124	1.563
Curve Fitting Toolbox	2.040	2.189	1.616

The response of the system plus the two models is shown in Fig. 11.

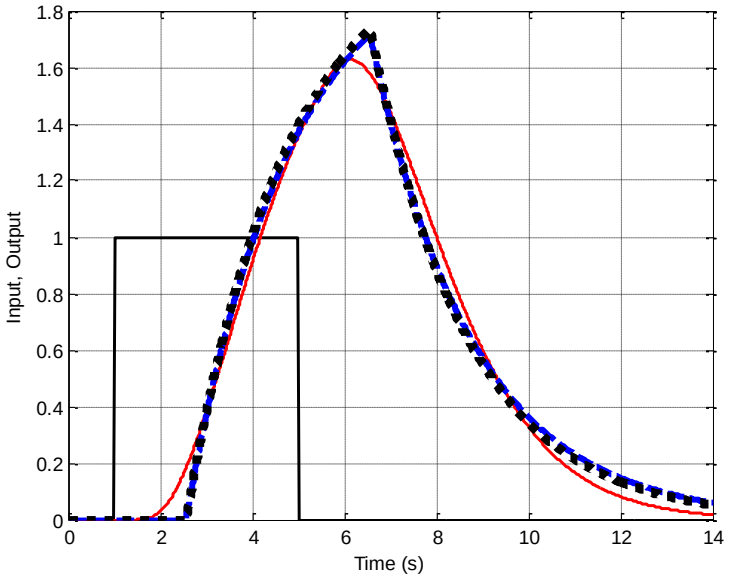

Fig. 11. Pulse responses: Solid = Input/3rd Order response; Dotted = IE model; Dashed = Curve Fitting Toolbox model

4.2 Coupled Tanks apparatus

The Coupled Tanks apparatus consists of two separate vertical tanks, as shown in Fig. 12. The tanks are connected by pipe containing a rotary valve, which allows the user to vary the flow characteristic between them. Each tank also has a drain pipe at its base, with a rotary valve that allows direct variable discharge into a reservoir. Liquid level is measured using a pressure sensing transducer, which is part of a simple bridge circuit providing a 0 to +10V signal. This corresponds to the 0 to 250mm scale mounted on the front panel adjacent to each tank window. The unit has two pumps, each driven by a 0 to +10V signal, which is the input to a small variable speed DC motor. An output voltage in the range 0 to +10V is available that is proportional to the pump flow rate. The aim of the identification experiment was to obtain a FOPDT transfer function relating the signal applied to the pump in tank 1 to the signal corresponding to the liquid level in tank 2.

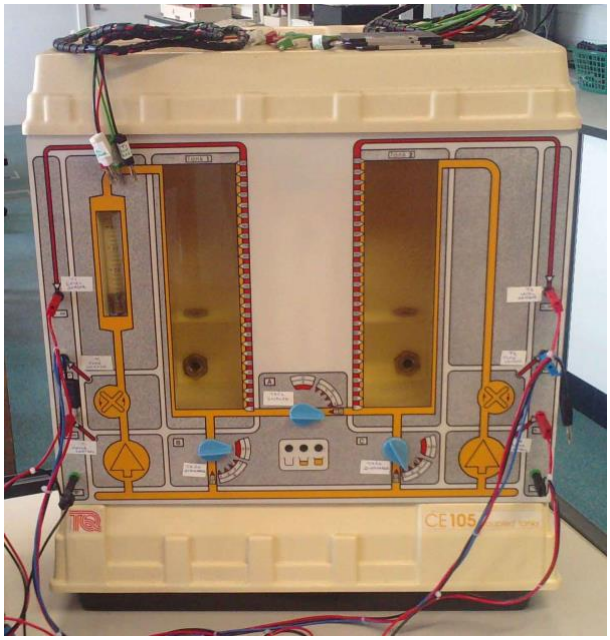

Fig. 12. Coupled Tanks apparatus

The first step was to collect the raw data for a step input. Using a sample time of 1s, 3400 data samples were collected. The results are shown in Fig. 13.

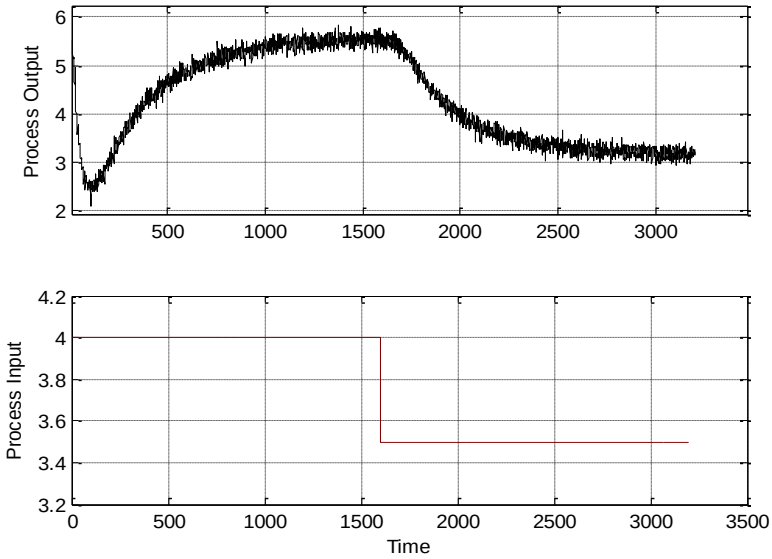

Fig. 13 Raw data collected from two-tank system

It is apparent from Fig. 13 that the process output was significantly contaminated by noise. It is also clear that the data needed to be cropped, in order to remove the transient behaviour prior to the step input and thereby enable the step response identification methods to be used effectively. The cropped data is shown in Fig. 14.

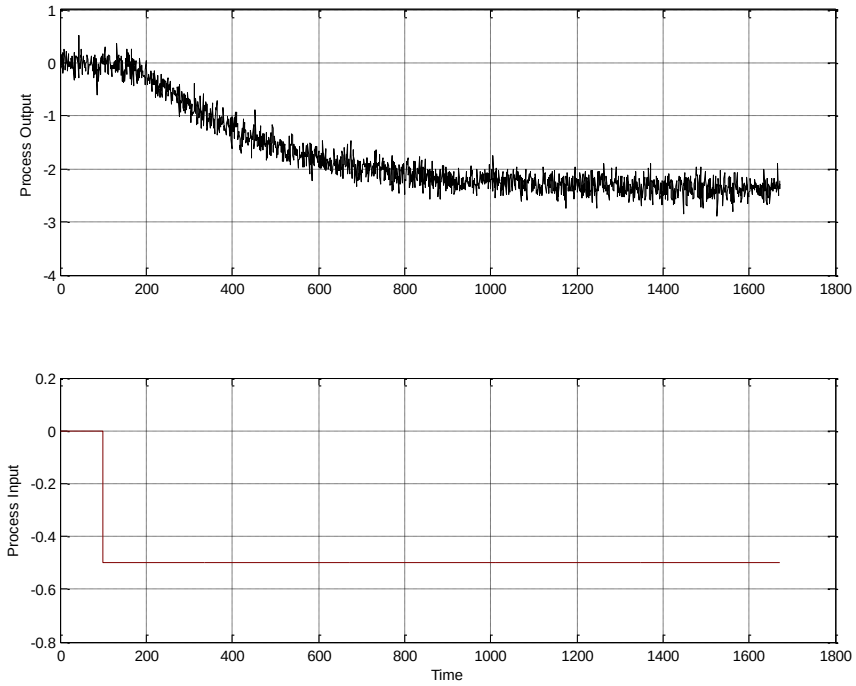

Fig. 14. Cropped data used by identification software

The parameters identified by the IE method, the Curve Fitting Toolbox, and the EzyFit Toolbox are shown in Table 3. A visual representation of the 'closeness of fit' using the parameters derived by the IE method and the EzyFit Toolbox is shown in Fig. 15.

Table 3
Two Tanks system, step response: FOPDT parameters

Identification method	K	T	L
IE method	4.60	257.2	106.5
Curve Fitting Toolbox	4.70	298.0	84.95
Ezyfit Toolbox	4.55	213.2	103.9

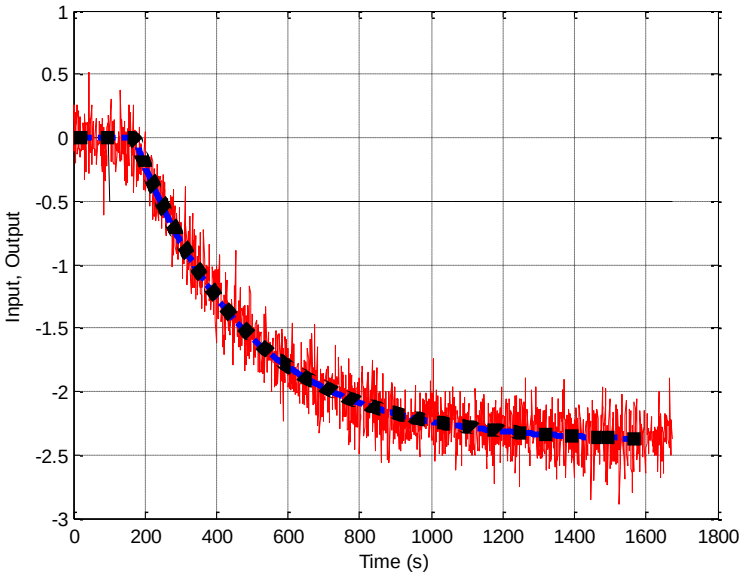

Fig. 15. Step responses: Solid = Two Tanks Input/Order response; Dotted = IE model; Dashed = EzyFit Toolbox model

5. Discussion and Conclusions

Several curve fitting software methods have been described, which enable the parameters of a FOPDT model to be determined when the system input is either a step function or a simple pulse. Their application has been demonstrated with examples where the identification data was derived either by simulation, or acquired from an experimental process. The effectiveness of the different methods has been demonstrated and the relative advantages and disadvantages of each discussed.

All of the methods worked well with simulated FOPDT data, where perfect (or almost perfect) results were obtained under controlled, noise-free conditions. They also appeared to work well both with the simulated third order system, and with the experimental data. In the case of the simulated FOPDT data this was easy to assess, simply by comparing the derived FOPDT parameters with the simulation values. In the latter two cases, however, performance assessment was largely qualitative in nature and achieved by superimposing step and pulse responses from the derived FOPDT models onto the original simulated higher order responses or experimental data.

A more quantifiable measure of performance with real system data can be achieved in an indirect way and this is the subject of continued study. This involves designing a range of PID-type controllers using the identified FOPDT parameters and measuring their closed loop performance. A more rigorous study to quantify the effects of varying noise, sampling times, data collection times, and other factors that affect the performance of these and other identification methods is also underway. A longer term goal is to develop a systematic and semi-automated approach to system identification, both for FOPDT and higher order models. This will include guidelines for pre- and post-processing of acquired data, with a view to optimising performance.

The results presented here, however, demonstrate the potential value of free or low cost, user-friendly, time domain-based identification methods that require only limited specialist knowledge to implement and employ.

6. References

[1] J.G. Ziegler, N.B. Nichols, Optimum settings for automatic controllers, Trans ASME, 65 (1942) 433-444

[2] G.H. Cohen, G.A. Coon, Theoretical considerations of retarded control, Trans ASME,75, 827-834,1953

[3] W.S. Levine (Editor), The Control Handbook: 2nd Edition, CRC Press, 2011.

[4] R.C. Oldenbourg, H. Sartorius, The dynamics of automatic control, ASME, New York, 1948.

[5] J.W. Sten, Evaluating second-order parameters, Instrumentation Technology, 17 (1970) 39-41.

[6] H. Rake, Step response and frequency response methods, Automatica, 16 (1980) 519-526.

[7] Y. Nishikawa, N. Sannomiya, T. Ohata, H. Tanka, A method for auto-tuning of PID control parameters, Automatica, 20 (1984) 321-332.

[8] K.J. Astrom, T. Hagglund, PID controllers: Theory, design and tuning, Instrument Society of America, 1995.

[9] Q. Bi, W. Cai, E. Lee, Q.G. Wang, C.C. Hang, Y. X. Zhang, Robust identification of first-order plus dead-time model from step response, Control Engineering Practice, 7(1) (1999) 71-77

[10] Q.G. Wang, Y. Zhang, Robust identification of continuous systems with dead-time from step response, Automatica, 37(3) (2001) 377-390

[11] S. Ahmed, B. Huang, S.L. Shah, Identification from step responses with transient initial conditions, Journal of Process Control, 18(2) (2008) 121-130

[12] M. Liu, Q.G. Wang, B. Huang, C.C. Hang, Improved identification of continuous-time delay processes from piecewise step test, Journal of process Control, 17(1) (2007) 51-57

[13] S. Ahmed, Process identification using non-ideal steps, 9th Int. Symp. On Dynamics and Control of Process Systems (DYCOPS, 2010), Leuven, Belgium, July 5-7, (2010)

[14] L. Ljung, Linear system identification as curve fitting, Linkoping University, Sweden, 12/2002

[15] P. C. Young, An instrumental variable method for real-time identification of a noisy process, Automatica, 6/2 (1970), 271-287.

[16] K. Burn, L. Maerte, C.S. Cox, 2010. A MATLAB toolbox for teaching modern system identification methods for industrial process control. International Journal of Mechanical Engineering Education, 38 (4), pp. 352-264.

[17] S. Ahmed, B. Huang, S.L. Shah, Novel identification method from step response, Control Engineering Practice, 15(5) (2007) 545-556.

[18] J.P. Norton, An introduction to identification, Dover Publications,2009

[19] Michigan Tech Open Sustainability Lab. Available <http://www.appropedia.org/Curve_fitting_to_a_set_of_data>.

[20] H.P. Gavin, The Levenberg-Marquardt method for nonlinear squares curve fitting problems. Available <http://people.duke.edu/~hpgavin/ce281/lm.pdf>.

[21] MATLAB Curve Fitting Toolbox: User's Guide, Mathworks Inc., 2013.

[22] MATLAB Optimization Toolbox: User's Guide. Mathworks Inc., 2009.

[23] Ezyfit : A free curve fitting toolbox for MATLAB. Available <http://www.fast.u-psud.fr/ezyfit>.

[24] G. Fedele, A new method to estimate a first-order plus time-delay mode from step response, J. Franklin Inst. 346 (2009) 1-9

[25] A. O'Dwyer, Handbook of PI and PID controller tuning rules, Imperial College Press, London, 2006.